Министерство науки и высшего образования РФ

ФГАОУ ВО Пермский национальный исследовательский

политехнический университет

Кафедра «Вычислительная математика, механика и биомеханика»

Отчет по лабораторной работе № 6

тема «Циклы»

по дисциплине «Информатика»

Выполнил: студент группы ИСТ-21-1б Евсеев В.С.

Проверил: доцент каф. ВММБ Каменских А.А.

Пермь, 2021

Содержание

[Задание 1 4](#_Toc91367013)

[1.1. Общая постановка задачи 4](#_Toc91367014)

[1.2. Решение задачи 4](#_Toc91367015)

[1.3. Тестирование программы с проверкой 4](#_Toc91367016)

[Задание 2 7](#_Toc91367017)

[2.1. Общая постановка задачи 7](#_Toc91367018)

[2.2. Решение задачи 7](#_Toc91367019)

[2.3 Тестирование программы с проверкой 7](#_Toc91367020)

[Задание 3 9](#_Toc91367021)

[3.1. Общая постановка задачи 9](#_Toc91367022)

[3.2. Решение задачи 9](#_Toc91367023)

[3.2.1. Решение первого выражения 9](#_Toc91367024)

[3.2.2 Решение второго выражения 9](#_Toc91367025)

[3.2.3 Решение третьего выражения 9](#_Toc91367026)

[3.3. Тестирование с проверкой 10](#_Toc91367027)

[3.3.1. Тестирование с проверкой первого выражения 10](#_Toc91367028)

[3.3.2. Тестирование с проверкой второго выражения 10](#_Toc91367029)

[3.3.3. Тестирование с проверкой третьего выражения 12](#_Toc91367030)

[Задание 4 14](#_Toc91367031)

[4.1. Общая постановка задачи 14](#_Toc91367032)

[4.2. Решение задачи 14](#_Toc91367033)

[4.3. Тестирование с проверкой 15](#_Toc91367034)

[Задание 5 18](#_Toc91367035)

[5.1. Общая постановка задачи 18](#_Toc91367036)

[5.2. Решение задачи 18](#_Toc91367037)

[5.3. Тестирование программы с проверкой 18](#_Toc91367038)

[Задание 6 22](#_Toc91367039)

[6.1. Общая постановка задачи 22](#_Toc91367040)

[6.2. Решение задачи 22](#_Toc91367041)

[6.3. Проверка с тестированием 23](#_Toc91367042)

[Задание7 25](#_Toc91367043)

[7.1. Общая постановка задачи 25](#_Toc91367044)

[7.2. Решение задачи 25](#_Toc91367045)

[7.3. Тестирование с проверкой 25](#_Toc91367046)

[Задание 8 27](#_Toc91367047)

[8.1. Общая постановка задачи 27](#_Toc91367048)

[8.2. Решение задачи 27](#_Toc91367049)

[8.3. Тестирование работы программы с проверкой 28](#_Toc91367050)

[Задание 9 31](#_Toc91367051)

[9.1. Общая постановка задачи 31](#_Toc91367052)

[9.2. Решение задачи 31](#_Toc91367053)

[9.3. Тестирование с проверкой 32](#_Toc91367054)

[Проверка №1 33](#_Toc91367055)

[Проверка №2 35](#_Toc91367056)

[Проверка №3 36](#_Toc91367057)

[Блок-схема 39](#_Toc91367058)

[Задание 10 41](#_Toc91367059)

[10.1. Общая постановка задачи 41](#_Toc91367060)

[10.2. Решение задачи 41](#_Toc91367061)

[10.3. Тестирование с проверкой 42](#_Toc91367062)

# Задание 1

## 1.1. Общая постановка задачи

Найти сумму первых N членов ряда и найти сумму членов ряда, которые меньше заданного с клавиатуры числа M:

**

## 1.2. Решение задачи

import java.util.Scanner;  
  
import static java.lang.System.*out*;  
  
  
public class Task61 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 *out*.println("Введите количество членов ряда n:");  
 int n = scanner.nextInt();  
 *out*.println("Введите число m:");  
 double m = scanner.nextDouble();  
 *out*.println("Введите константу x:");  
 double x = scanner.nextDouble();  
 double Sn = 0;  
 double Sm = 0;  
 for (int i = 0; i <= n; i++) {  
 Sn += Math.*pow*(x, i) / *Fact*(i);  
 if ((Math.*pow*(x, i) / *Fact*(i)) < m) {  
 Sm += Math.*pow*(x, i) / *Fact*(i);  
 }  
 }  
 *out*.println("Сумма первых членов ряда n: " + Sn);  
 *out*.println("Сумма членов ряда, которые меньше заданного с клавиатуры числа m: " + Sm);  
 }  
 public static int Fact(int n){  
 int result = 1;  
 if (n == 0) {  
 return result;  
 } else {  
 for (int i = 1; i <= n; i++) {  
 result \*= i;  
 }  
 return result;  
 }  
 }  
}

## 1.3. Тестирование программы с проверкой

Для проверки в MS Excel создана таблица, в которой в ячейку B1 записано значение n, в ячейку B2 значение m, в ячейку B3 значение x, в ячейки A5:A54 значение i, в ячейки B5 функция для нахождения суммы членов ряда, в ячейки C5 функция для нахождения членов ряда, которые меньше заданного с клавиатуры числа m, в ячейку B6 сумма текущей функции для нахождения суммы членов ряда с предыдущей, C6 сумма текущей функция для нахождения членов ряда, которые меньше заданного с клавиатуры числа m с предыдущей.

B5)=$B$3^A5/ФАКТР(A5)

C5)=ЕСЛИ(B5<$B$2;$B$3^A5/ФАКТР(A5);0)

B6)=$B$3^A6/ФАКТР(A6)+B5

C6)=ЕСЛИ($B$3^A5/ФАКТР(A5)<$B$2;$B$3^A5/ФАКТР(A5);0)+C5

На рис. 1 приведён вид решения в MS Excel.

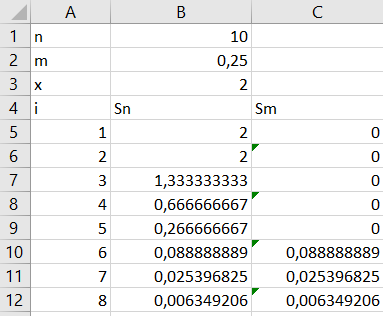


Рисунок 1. Вид решения в MS Excel задания 1

В таблице 1 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением задачи в MS Excel.

Таблица 1

Тестирование работы программы и проверка решения задачи 1

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Решение MS Excel |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 2

## 2.1. Общая постановка задачи

Билет называют «счастливым», если в его номере сумма первых трех цифр равна сумме последних трех (цифры шестизначные). Подсчитать число «счастливых» билетов, которые есть в катушке у кондуктора, если на начало рабочего дня начальный номер катушки N (минимальный номер билета 000000, максимальный номер билета 999999).

## 2.2. Решение задачи

import java.util.Scanner;  
  
import static java.lang.System.*out*;  
  
public class Task62 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 *out*.println("Введите начальный номер билета:");  
 int n = scanner.nextInt();  
 int k = 0;  
 for (int i = n; i <= 999999; i++) {  
 if ( *firstThree*(i) ==*secondThree*(i)) {  
 k++;  
 }  
 }  
 *out*.println("Число счастливых билетов: " + k);  
 }  
  
 public static int firstThree(int n){  
 int a = n % 1000;  
 a = (a % 10) + (a / 100) + ((a / 10) % 10);  
 return a;  
 }  
  
 public static int secondThree(int n){  
 int b = n / 1000;  
 b = (b % 10) + (b / 100) + ((b / 10) % 10);  
 return b;  
 }  
}

## 2.3 Тестирование программы с проверкой

В таблице 2 представлено тестирование работы программы на языке Java.

Таблица 2

Тестирование работы программы на языке Java для задачи 2

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |
| 2 |  |
| 3 |  |

-

+

-

+

n, k=0, i=n



Цикл for



Число счастливых билетов k

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и блок-схемы совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 3

## 3.1. Общая постановка задачи

Написать программы, которые вычисляют выражения:

, , 

## 3.2. Решение задачи

### 3.2.1. Решение первого выражения

import static java.lang.System.*out*;  
  
import static java.lang.Math.\*;  
  
public class Task631 {  
 public static void main(String[] args) {  
 *out*.println("Сумма сумм равна: " + *summSumm*());  
 }  
 public static int summSumm() {  
 int summ = 0;  
 for (int i = 1; i <= 8; i++) {  
 for (int j = 1; j <= i; j++){  
 summ += *pow*((j + i), 2);  
 }  
 }  
 return summ;  
 }  
  
}

### 3.2.2 Решение второго выражения

import static java.lang.System.*out*;  
  
public class Task632 {  
 public static void main(String[] args) {  
 *out*.println("Произведение произведений равно: " + *multiplie*());  
 }  
 public static int multiplie() {  
 int multi = 1;  
 for (int i = 1; i <= 5; i++) {  
 for (int j = 1; j <= i; j++) {  
 multi \*= j;  
 }  
 }  
 return multi;  
 }  
}

### 3.2.3 Решение третьего выражения

import static java.lang.System.*out*;  
  
public class Task633 {  
 public static void main(String[] args) {  
 *out*.println("Сумма произведений : " + *summMultiMultiplie*());  
 }  
 public static int summMultiMultiplie() {  
 int summ = 0;  
 int multik = 1, multij = 1;  
 for (int i = 1; i <= 8; i++) {  
 for (int j = i; j <= 8; j++) {  
 for (int k = 1; k <= 2 \* i; k++) {  
 multik \*= 2 \* i \* j - k;  
 }  
 multij \*= multik;  
 multik =1;  
 }  
 summ += multij;  
 multij = 1;  
 }  
 return summ;  
 }  
}

## 3.3. Тестирование с проверкой

### 3.3.1. Тестирование с проверкой первого выражения

Для проверки в MS Excel создана таблица, в которой в ячейки B1-I1 записаны значения j, в ячейки A2:A9 значения i, в ячейки J2;J9 сумма строк, в ячейку J10 сумма ячеек J2;J9, в ячейки B2:B9, C3:C9, D4:D9, E5:E9, F6:F9, G7:G9, H8:H9, I9 квадрат суммы i и j.

В таблице 3 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением задачи в MS Excel.

Таблица 3

Тестирование работы программы и проверка решения задачи 3.1

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Решение MS Excel |
| 1 |  |  |

Сравнение решения задачи с использованием одного прикладного пакета показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

### 3.3.2. Тестирование с проверкой второго выражения

Для проверки в MS Excel создана таблица, в которой в ячейки B1-F1 записаны значения j, в ячейки A2:A6 значения i, в ячейки J2:J9 произведение строк, в ячейку G7 произведение ячеек G2:G6, в ячейки B2:B6, C3:C6, D4:D6, E5:E6, F6 значение чисел j.

В таблице 4 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением задачи в MS Excel.

Таблица 4

Тестирование работы программы и проверка решения задачи 3.2

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Решение MS Excel |
| 1 |  |  |

Сравнение решения задачи с использованием одного прикладного пакета показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

### 3.3.3. Тестирование с проверкой третьего выражения

Для тестирования программы в MS Excel была создана таблица, в которой в ячейки C3:C4 записаны начальные и конечные значения i, в ячейки D3:D4 - j, в ячейки E3:E4 - k, в таблице A6:C8 записаны уже высчитанные значения j и k при i=1, в таблице E6:I8 – при i=2. В таблице A10:C12 в ячейках A12 и B12 приведено значение выражения  для k=1 и k=2 соответственно, j=1, i=1, в ячейке С12 записано произведение двух полученных чисел. Аналогично для таблицы A14:C16 только уже при j=2. В ячейку B19 записан результат произведения чисел, полученных в итоге в таблице A10:C12 и A14:C16. В таблице E10:I12 в ячейках E12:H12 приведено значение произведения выражения  для k=1, 2, 3, 4 соответственно, i=2, j=2, в ячейку I12 записано произведение полученных чисел. В ячейке F19 записан результат произведения, полученный в таблице E12:H12. В ячейке E21 записан результат суммы произведений при i=1 и i=2.

На рис. 2 представлено решение Java для конечного значения i и j равного 8.



Рис.2. Решение Java для i<=8 и j<=8

В таблице 5 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением задачи в MS Excel.

Таблица 5

Тестирование работы программы и проверка решения задачи 3.3

|  |  |  |
| --- | --- | --- |
| N п/п | Решение Java | Решение MS Excel |
| 1 |  |  |

Сравнение решения задачи с использованием одного прикладного пакета показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 4

## 4.1. Общая постановка задачи

Пользователь вводит целое десятичное число. Написать программу перевода целого десятичного числа из десятичной системы счисления в систему счисления с основанием, которое пользователь вводит с клавиатуры (вводим ограничения на системы счисления, в которых есть буквенное обозначение цифр, так же не может быть введено 0 и 1 как основание системы счисления). После перевода сделать проверку. Пользователь может выбирать систему счисления до бесконечности, т.е. необходимо предусмотреть внешний цикл с вопросом к пользователю о необходимости продолжать перевод из одной системы счисления в другую.

## 4.2. Решение задачи

import java.util.Scanner;  
  
import static java.lang.System.*out*;  
  
public class Task64 {  
  
 public static void main(String[] args){  
 *Scan*();  
 }  
 public static int Scan(){  
 Scanner scanner = new Scanner(System.*in*);  
 try{  
 *out*.println("Введите целое десятичное число");  
 int input = scanner.nextInt();  
 *out*.println("Введите основание системы счисления");  
 int base = scanner.nextInt();  
 if (base < 2 | base > 10){  
 *out*.println("Ошибка");  
 return *Scan*();  
 }  
 else {  
 *Tranfer*(input, base);  
 }  
 }  
 catch (Exception e){  
 *out*.println("Ошибка");  
 return *Scan*();  
 }  
 return 0;  
 }  
 public static int Tranfer(int input, int base){  
 String resultStr = "";  
 while (input > 0){  
 resultStr += (input % base);  
 input /= base;  
 //System.out.println(input);  
 }  
 int result = Integer.*valueOf*(resultStr);  
 *out*.println("Результат перевода = " + result);  
 return *NewScan*(result, base);  
 }  
 public static int NewScan(int result, int oldBase){  
 Scanner scanner = new Scanner(System.*in*);  
 *out*.println("Перевести в следующую систему счисления?" +  
 "\nВведите 1, если да, иначе введите любой другой символ");  
 try{  
 int answer = scanner.nextInt();  
 if(answer == 1){  
 *out*.println("Введите новую систему счисления");  
 int newBase = scanner.nextInt();  
 if (newBase < 2 | newBase > 10){  
 *out*.println("Ошибка");  
 return *NewScan*(result, oldBase);  
 }  
 else {  
 *TransferToNewBase*(result, oldBase, newBase);  
 }  
 }  
 }  
 catch (Exception e){  
 *out*.println("Ошибка");  
 return *NewScan*(result, oldBase);  
 }  
 return 0;  
 }  
 public static int TransferToNewBase(int number, int oldBase, int newBase){  
 int result = Integer.*parseInt*(String.*valueOf*(number), oldBase);  
 String resultStr = "";  
 while (result > 0){  
 resultStr += (result % newBase);  
 result /= newBase;  
 //System.out.println(input);  
 }  
 int resultEnd = Integer.*valueOf*(resultStr);  
 *out*.println("Результат перевода = " + resultEnd);  
 return *NewScan*(resultEnd, newBase);  
 }  
}

## 4.3. Тестирование с проверкой

В таблице 6 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением в онлайн калькуляторе.

Таблица 6

Тестирование работы программы и проверка решения задачи 4

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Онлайн калькулятор |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |

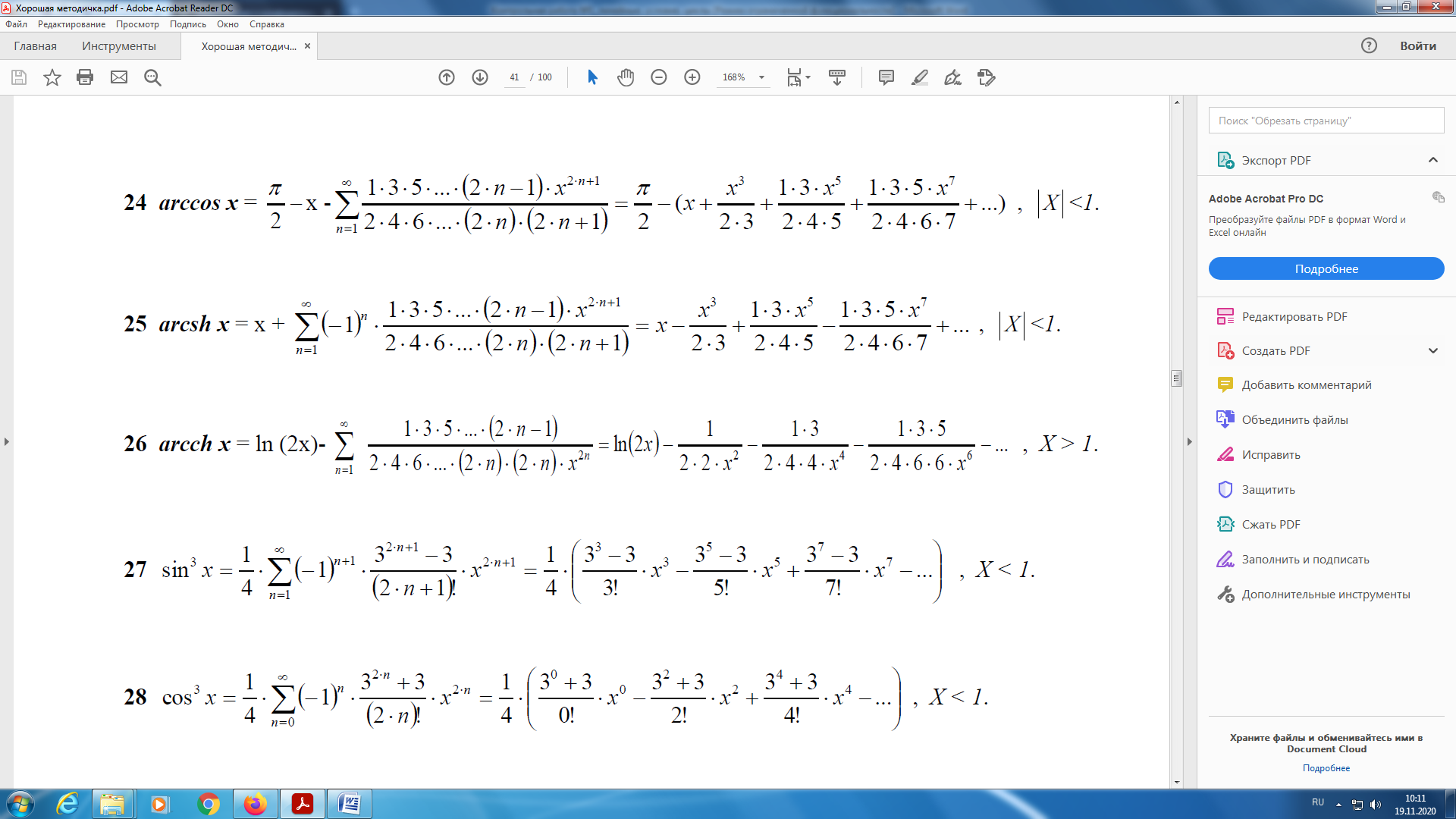
<https://calculatori.ru/perevod-chisel.html>

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и онлайн калькулятора совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 5

## 5.1. Общая постановка задачи

Организовать цикл для нахождения функции , через ряд Маклорена с погрешностью 10-2 – 10-4, остановка итерационной процедуры , где  – это погрешность. Вывести значение суммы ряда и итоговую погрешность для всех вариантов остановки итерационной процедуры.



## 5.2. Решение задачи

import static java.lang.System.*out*;  
  
import static java.lang.Math.\*;  
  
import java.util.Scanner;  
  
public class Task65 {  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
 *out*.println("Введите переменную x меньше 1:");  
 double x = scanner.nextDouble();  
 double Eps = *pow*(10, -2);  
 if (x >= 1) {  
 *out*.println("Ошибка" +  
 "\nВведите x меньше 1");  
 }  
 else {  
 while (Eps >= *pow*(10, -4)) {  
 double summ = 0;  
 int n = 1;  
 while ((*abs*(*pow*(*sin*(x), 3) - summ/4)) >= Eps) {  
 summ += *pow*(-1, (n + 1)) \* ((*pow*(3, (2 \* n + 1)) - 3) / *Fact*(2 \* n + 1)) \* *pow*(x, (2 \* n + 1));  
 n++;  
 }  
 *out*.printf("Значение ряда Маклорена при погрешности " + Eps + " равно " + "%.12f", (1.0 / 4) \* summ);  
 *out*.println("");  
 Eps \*= 0.1;  
 }  
 }  
 }  
 public static int Fact(int n) {  
 int result = 1;  
 for (int j = 1; j <= n; j++) {  
 result \*= j;  
 }  
 return result;  
 }  
}

## 5.3. Тестирование программы с проверкой

Для поверки в MS Excel создана таблица, в которой в ячейку B1 записано значение x, в ячейку B2 значение , в ячейку B3 значение , в ячейки A5:A34 значение i, в ячейки B5:B34 функция для нахождения значения ряда Маклорена, в ячейки C5:C34, в ячейки С5:С34 функция для нахождения суммы ряда Маклорена.

B5)=1/4\*(-1)^(A5+1)\*(3^(2\*A5+1)-3)/ФАКТР(2\*A5+1)\*$B$1^(2\*A5+1)

C5)=B5

C6)=ЕСЛИ(ABS($B$2-C5)>=$B$3;C5+B6;"Otvet vishe")

На рис.3 представлен вид решения в MS Excel.

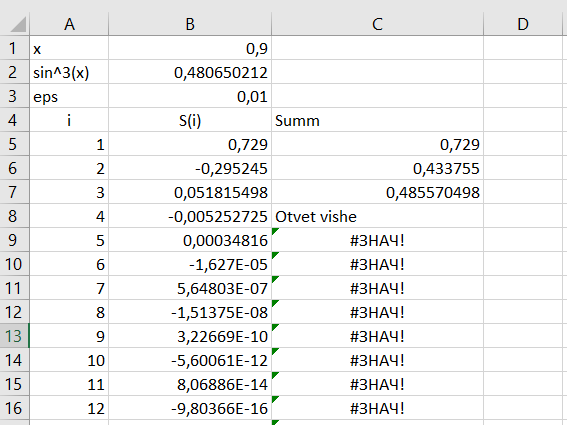


Рисунок 3. Вид решения в MS Excel задания 5

В таблице 7 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением задачи в MS Excel.

Таблица 7

Тестирование работы программы и проверка решения задачи 5

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Решение |
| 1 |  |  |
|  |
|  |
| 2 |  |  |
|  |
|  |
| 3 |  |  |
|  |
|  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 6

## 6.1. Общая постановка задачи

Переделать программу (класс) задания 6 лабораторной работы 5 о попадании точки в область в класс без метода main с названием Oblast. Метод main заменить на метод Oblast c входными данными в виде координат произвольной точки пространства и выходными данными типа boolean (true – если точка попала в область, false – если точка не попала в область). Создать программу, взаимодействующую с классом Oblast (без использования наследования), в которой пользователь в цикле проверяет попадание точек в область до бесконечности, т.е. необходимо предусмотреть цикл с вопросом к пользователю о необходимости проверки точки.

## 6.2. Решение задачи

import static java.lang.Math.\*;  
  
import static java.lang.System.*out*;  
  
import java.util.Scanner;  
  
public class Task66 {  
 public boolean Oblast(double x, double y) {  
 int Count1 = 0;  
 double Sqrt1 = *sqrt*(1 - *pow*((x - 2), 2));  
 double Sqrt2 = *sqrt*(1 - *pow*((x + 2), 2));  
 //Правая часть  
 if ((x >= 0) & (x <= 6) & (y >= -3.0 / 2 \* x + 4) & (y >= y / 3 + 1.0 / 3) &  
 (y >= -2.0 / 7 \* x + 10.0 / 7) &  
 (y >= 0) & (y <= *sqrt*(36 - x \* x)) & ((y >= Sqrt1 + 4) | (y <= -Sqrt1 + 4))) {  
 Count1++;  
 }  
 //Левая часть  
 if ((x >= -5) & (x <= 0) & (y <= 4) & (y <= 4.0 / 3 \* x + 20.0 / 3) &  
 (y >= 0) & (y >= x / 2 +1) & ((y >= Sqrt2 + 2) | y <= -Sqrt2 + 2)) {  
 Count1++;  
 }  
 return (Count1 > 0);  
 }  
}  
class HitChecker {  
 public static void main(String[] args) {  
 *Scan*();  
 }  
 public static double Scan() {  
 try {  
 Scanner scanner = new Scanner(System.*in*);  
 *out*.println("Введите значение координаты x:");  
 double x = scanner.nextDouble();  
 *out*.println("Введите значение координаты y:");  
 double y = scanner.nextDouble();  
 Task66 hitChecker = new Task66();  
 *out*.println(hitChecker.Oblast(x, y));  
 *Cycle*();  
 }  
 catch (Exception e) {  
 *out*.println("Ошибка");  
 return *Scan*();  
 }  
 return 0;  
 }  
 public static double Cycle() {  
 Scanner scanner = new Scanner(System.*in*);  
 *out*.println("Вы хотите проверить еще одну точку?" +  
 "\nЕсли да, то введите 1, если нет - любой другой символ");  
 try {  
 int j = scanner.nextInt();  
 if (j == 1) {  
 *Scan*();  
 }  
 }  
 catch (Exception e) {  
 *out*.println("Ну ладно(");  
 }  
 return 0;  
 }  
  
}

## 6.3. Проверка с тестированием

-

+

-

+



Проверка на попадание точки в область фигуры

true

false



Ну ладно(

В таблице 8 представлено тестирование работы программы на языке Java.

Таблица 8

Тестирование работы программы на языке Java для задачи 6

|  |  |
| --- | --- |
| N п/п | Решение Java |
| 1 |  |

Сравнение решения задачи с использованием одного прикладного пакета показала, что решения задачи в Java и блок-схемы совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание7

## 7.1. Общая постановка задачи

Напечатать полную таблицу умножения в виде:

|  |  |  |  |
| --- | --- | --- | --- |
| 1 х 1 = 1 | 1 х 2 = 2 | ... | 1 х 9 = 9 |
| 2 х 1 = 2 | 2 х 2 = 4 | ... | 2 х 9 = 18 |
| ... | ... | ... | ... |
| 9 х 1 = 9 | 9 х 2 = 18 | ... | 9 х 9 = 81 |

## 7.2. Решение задачи

import static java.lang.System.*out*;  
  
public class Task67 {  
 public static void main(String[] args) {  
 *Multiplie*();  
 }  
 public static int Multiplie() {  
 int i, j;  
 for (i = 1; i <= 9; i++) {  
 for (j = 1; j <= 9; j++) {  
 *out*.print(i + " x " + j + " = " + (i \* j) + "\t");  
 }  
 *out*.println("");  
 }  
 return 0;  
 }  
}

## 7.3. Тестирование с проверкой

Цикл for

-

+

Цикл for

-

+

i=1, j=1

i<=9

j<=9

Вывод результата



j+1

Пустое слово

i+1

В таблице 9 представлено тестирование работы программы на языке Java.

Таблица 9

Тестирование работы программы на языке Java для задачи 7

|  |  |
| --- | --- |
| № п/п | Решение Java |
| 1 |  |

Сравнение решения задачи с использованием одного прикладного пакета показала, что решения задачи в Java и блок-схемы совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 8

## 8.1. Общая постановка задачи

Переделать программу (класс) задания 1 лабораторной работы 4 о нахождении значений 2-х функций в класс без метода main с названием FunctionMy. Метод main заменить на метод FunctionMy c входными данными. Создать программу, взаимодействующую с классом FunctionMy, в которой пользователь в цикле находит сумму 10 значений функции изменяя только один параметр функции в цикле, остальные параметры, которые входя в формулу, считаются константами. Взаимодействие с классом FunctionMy сделать в виде наследования.

## 8.2. Решение задачи

import java.util.Scanner;  
  
import static java.lang.System.*out*;  
  
public class Task682Cycle extends FunctionMy {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.*in*);  
 double x, h;  
 *out*.println("Введите начальное значение переменной x");  
 x = in.nextDouble();  
 *out*.println("Введите шаг для переменной h");  
 h = in.nextDouble();  
 double SummG = 0, SummU = 0;  
 for(int i = 0; i < 10; i++) {  
 if (*FunctionG*(x + i \* h) == *FunctionG*(x + i \* h))  
 SummG += *FunctionG*(x + i \* h);  
 if (*FunctionU*(x + i \* h) == *FunctionU*(x + i \* h))  
 SummU += *FunctionU*(x + i \* h);  
 }  
 *out*.println("SummG = " + SummG);  
 *out*.println("SummU = " + SummU);  
 }  
}

## 8.3. Тестирование работы программы с проверкой

Для проверки в MS Excel создана таблица, в которой в ячейке B1 записано значение y, в ячейке B2 - z, в ячейке B3 - y1, в ячейке B4 значение шага для h, в ячейке B5 - x0, в диапазон ячеек D2:D11 номер итерации i, в диапазон ячеек E2:E11 – значение x в каждой итерации, в диапазон ячеек F2:F11 значение функции G в каждой итерации, в диапазон ячеек G2:G11 значение функции U в каждой итерации, в ячейку E13 значение суммы функций G , в ячейку E14 – функций U.

F2)=(TAN(E2^4-6)-((COS($B$2+E2\*$B$1))^3))/((COS(E2^3+(SIN(ПИ()+E2))^3))^4)

G2)=(LN(E2-$B$3)+$B$3^4)/(EXP(1)^$B$3+2,355\*E2^2)

На рис. 4 представлен вид решения в MS Excel.

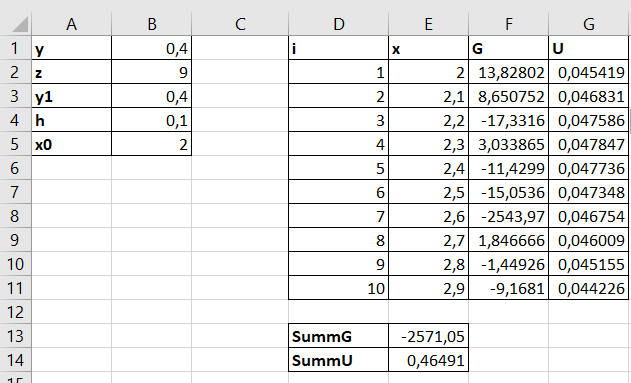


Рисунок 4. Вид решения в MS Excel задания 8

В таблице 7 представлено тестирование работы программы с проверкой решения задачи на языке Java с решением задачи в MS Excel.

Таблица 10

Тестирование работы программы и проверка решения задачи 8

|  |  |  |
| --- | --- | --- |
| № п/п | Решение Java | Решение MS Excel |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |

Сравнение решения задачи с использованием трех прикладных пакетов показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 9

## 9.1. Общая постановка задачи

Написать программу вычисление корня р-й степени (степень вводиться с клавиатуры) в рамках итерационной процедуры . Для определения используется итерационная процедура на основе формулы Ньютона , , при этом . Остановка итерационной процедуры , где  – точность вычисления. В рамках программы определить число итераций, которые потребовались для отыскания корня р-й степени в рамках цикла с параметром для точности от 10-2 до 10-6, шаг 10-1. Организовать форматированный вывод результатов в виде: Точность Корень Число итераций.

## 9.2. Решение задачи

import static java.lang.System.*out*;  
  
import static java.lang.Math.\*;  
  
import java.util.Scanner;  
  
public class Task69 {  
 public static void main(String[] args) {  
 *ScanP*();  
 }  
 public static double ScanP() {  
 try {  
 Scanner scannerP = new Scanner(System.*in*);  
 *out*.println("Введите корень p больше нуля:");  
 double p = scannerP.nextDouble();  
 if (p <= 0) {  
 *out*.println("Ошибка!");  
 *ScanP*();  
 }  
 else {  
 *ScanX*(p);  
 }  
 }  
 catch (Exception e) {  
 *out*.println("Ошибка!");  
 *ScanP*();  
 }  
 return 0;  
 }  
 public static double ScanX(double p) {  
 try {  
 Scanner scannerX = new Scanner(System.*in*);  
 *out*.println("Введите константу x больше нуля:");  
 double x = scannerX.nextDouble();  
 if (x <= 0) {  
 *out*.println("Ошибка!");  
 *ScanX*(p);  
 }  
 else {  
 *ScanY0*(p, x);  
 }  
 }  
 catch (Exception e) {  
 *out*.println("Ошибка!");  
 *ScanX*(p);  
 }  
 return 0;  
 }  
 public static double ScanY0(double p, double x) {  
 try {  
 Scanner scannerY0 = new Scanner(System.*in*);  
 double limitation = *pow*(*E*, ((*log*(x \* (p + 1))) / p));  
 *out*.println("Введите переменную y0 меньше " + limitation);  
 double y0 = scannerY0.nextDouble();  
 if (y0 >= limitation) {  
 *out*.println("Ошибка!");  
 *ScanY0*(p, x);  
 } else {  
 *Cycle*(p, x, y0);  
 }  
 }  
 catch (Exception e) {  
 *out*.println("Ошибка!");  
 *ScanY0*(p, x);  
 }  
 return 0;  
 }  
 public static double Cycle(double p, double x, double y0) {  
 try {  
 double Eps = *pow*(10, -2);  
 int n = 1;  
 while (Eps >= *pow*(10, -6)) {  
 double y = (1.0 / p) \* (((p - 1) \* y0) + (x / *pow*(y0, (p - 1))));  
 while (*abs*(y - y0) > Eps) {  
 y0 = y;  
 y = (1.0 / p) \* (((p - 1) \* y0) + (x / *pow*(y0, (p - 1))));  
 n += 1;  
 }  
 *out*.println("При точности " + Eps + " корень p-ой степени равен " + y +  
 ", число итераций равно " + n);  
 Eps \*= 0.1;  
 }  
 }  
 catch (Exception e) {  
 *out*.println("Ошибка!");  
 *Cycle*(p, x, y0);  
 }  
 return 0;  
 }  
}

## 9.3. Тестирование с проверкой

Для тестирования в MS Excel создана таблица, в которой в ячейку C1 записано значение степени корня p, в ячейку C2 значение подкоренного выражения x, в ячейку С3 начальное значение y0, в ячейку С4 функция, проверяющая условие на удовлетворение y0, в ячейку H1 – погрешность , в ячейку H3 функция для проверки вычисления корня, в ячейку B8 и далее вниз номер итерации n, в ячейку C8 функция, проверяющая подходит ли y0, в ячейку С9 функция вычисляющая корень p-ой степени, в ячейку C10 и далее вниз функция с условием, вычисляющая корень p-ой степени.

С4)=ЕСЛИ(C3<EXP(LN(C2\*(C1+1)));1;0)

H3)=СТЕПЕНЬ(C2;1/C1)

C8)=ЕСЛИ(C4=1;C3;"Введите другое y0")

C9)=1/$C$1\*(($C$1-1)\*C8+$C$2/C8^($C$1-1))

С10)=ЕСЛИ(ИЛИ(ЕСЛИ(C9<>"Решение найдено, см. выше";ABS(C9-C8)<=$H$1);C9="Решение найдено, см. выше");"Решение найдено, см. выше";1/$C$1\*(($C$1-1)\*C9+$C$2/C9^($C$1-1)))

На рис. 5 представлен вид решения в MS Excel.

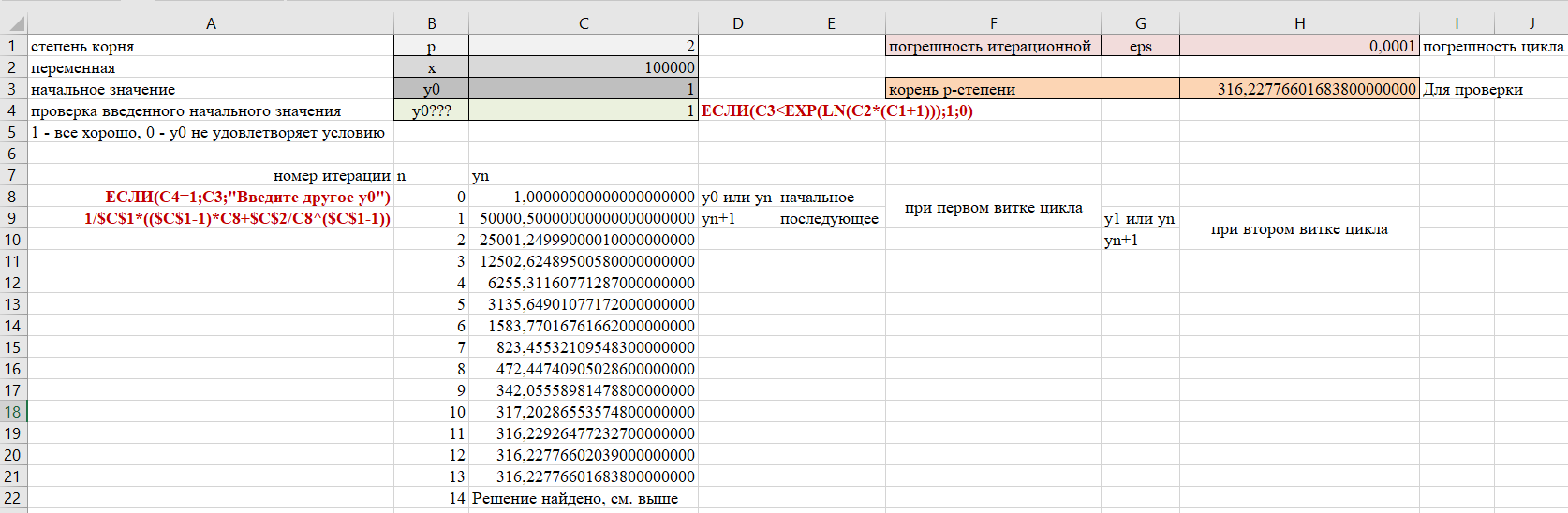


Рисунок 5. Вид решения в MS Excel задания 9

### Проверка №1

На рис. 6 представлен вид решения в Java.

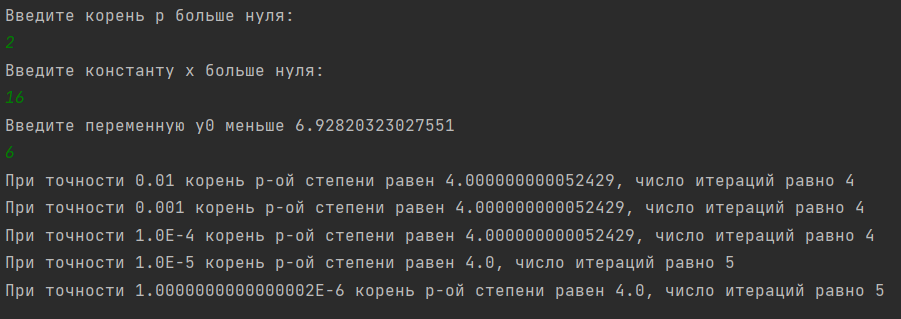


Рисунок 6. Вид решения Java задания 9 №1

На рис. 7-11 представлена проверка решения задания 9 в MS Excel.

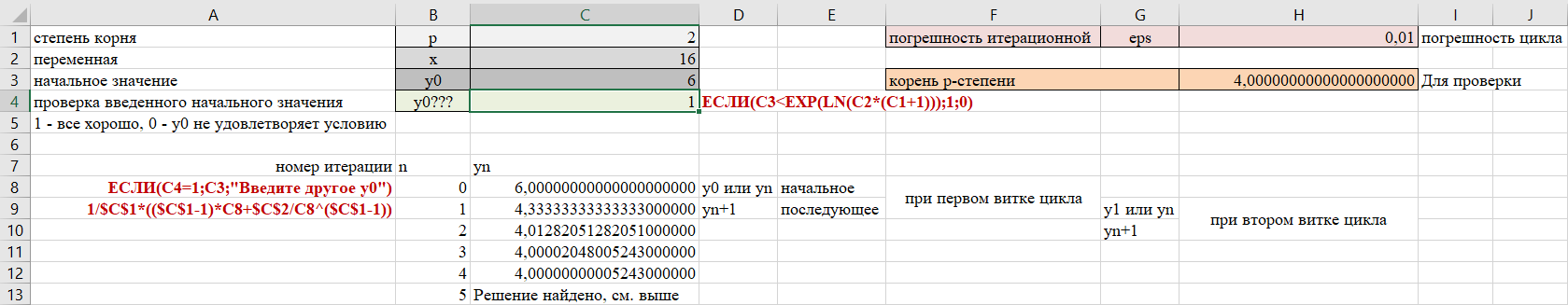


Рисунок 7. Проверка решения задания 9 при  №1

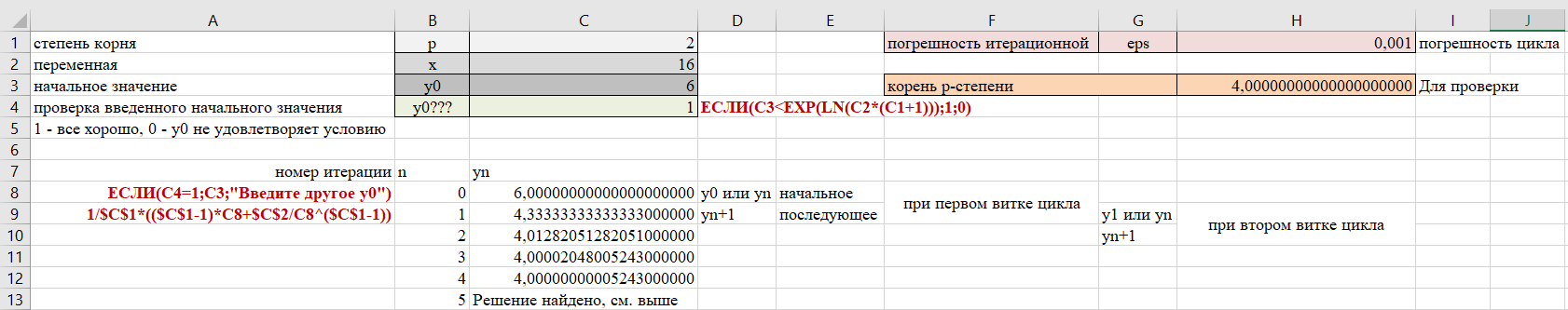


Рисунок 8. Проверка решения задания 9 при  №1

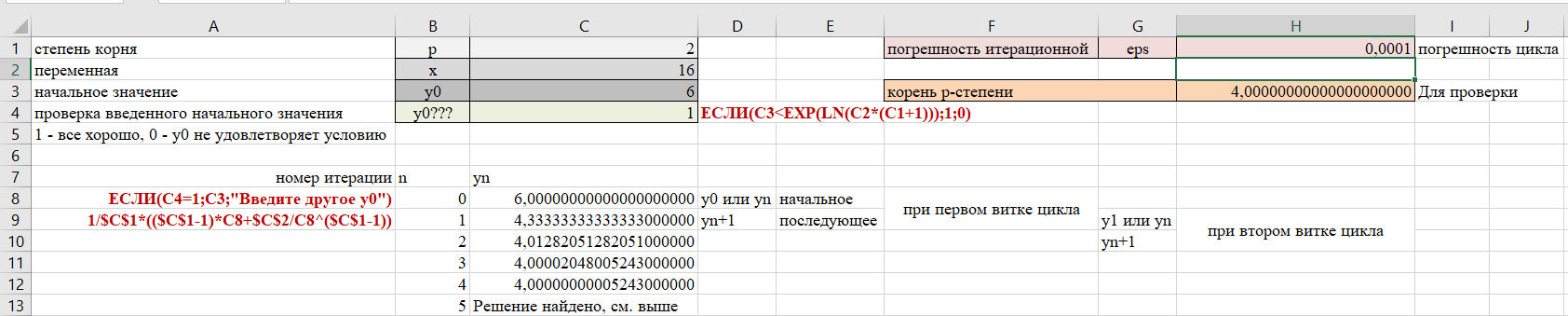


Рисунок 9. Проверка решения задания 9 при  №1

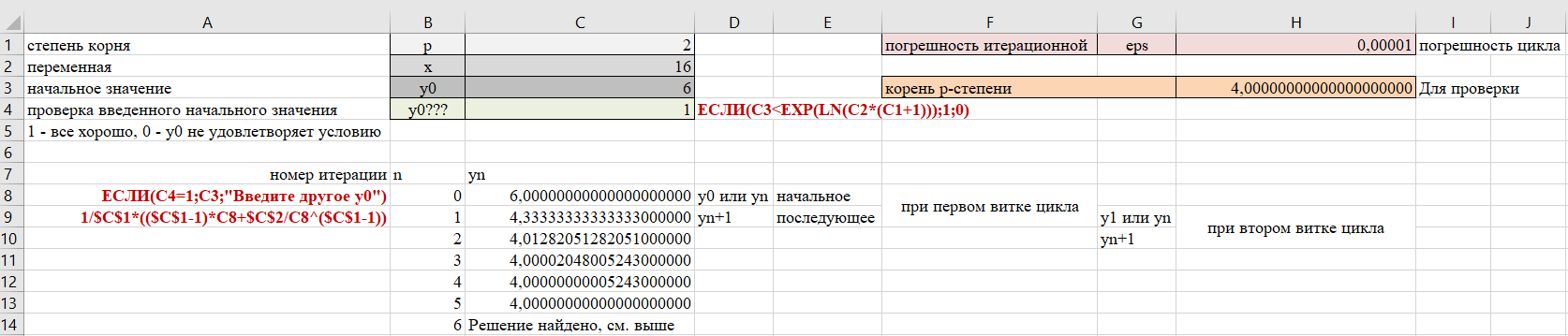


Рисунок 10. Проверка решения задания 9 при  №1

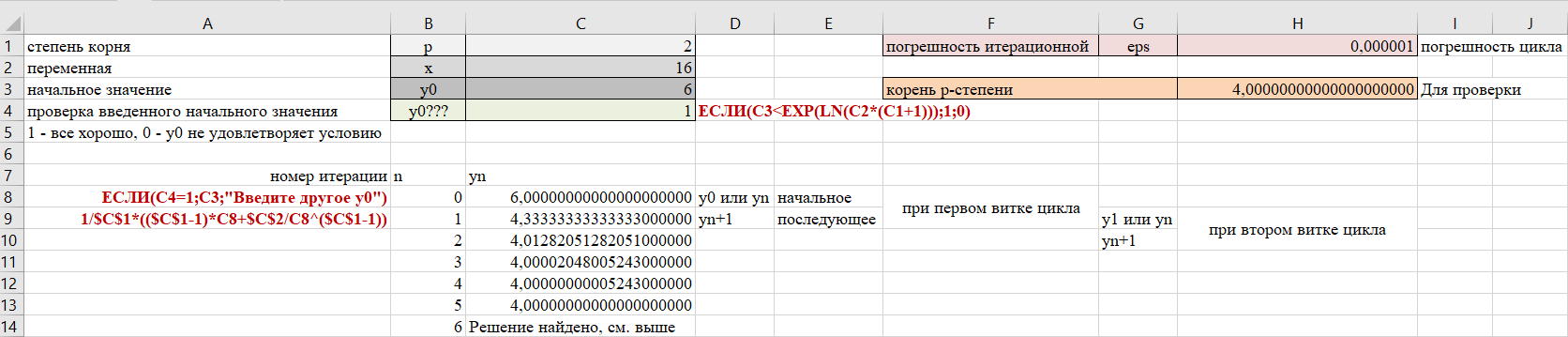


Рисунок 11. Проверка решения задания 9 при  №1

### Проверка №2

На рис.12 представлен вид решения в Java.

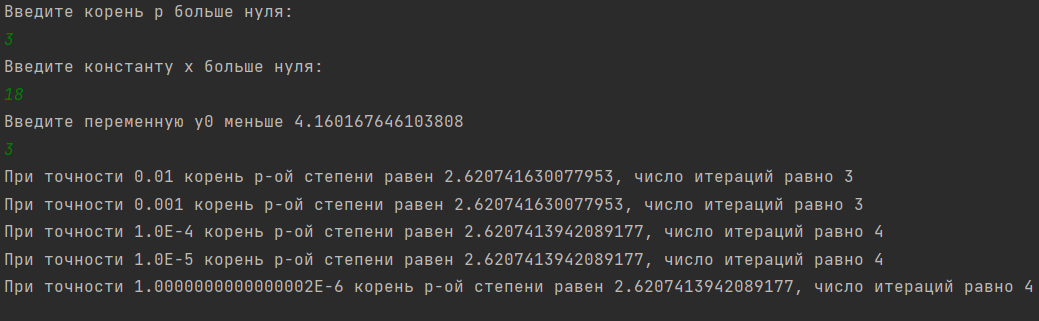


Рисунок 12. Вид решения Java задания 9 №2

На рис. 13-17 представлена проверка задания 9 в MS Excel.

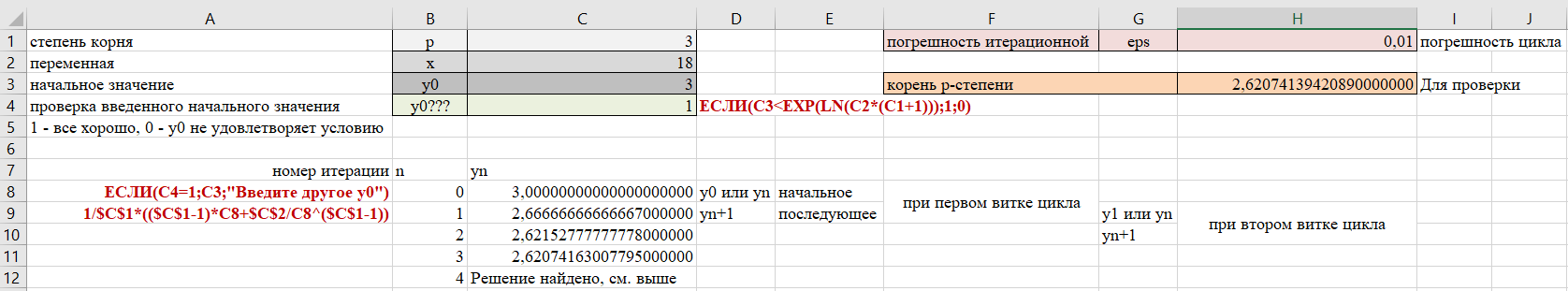


Рисунок 13. Проверка решения задания 9 при  №2

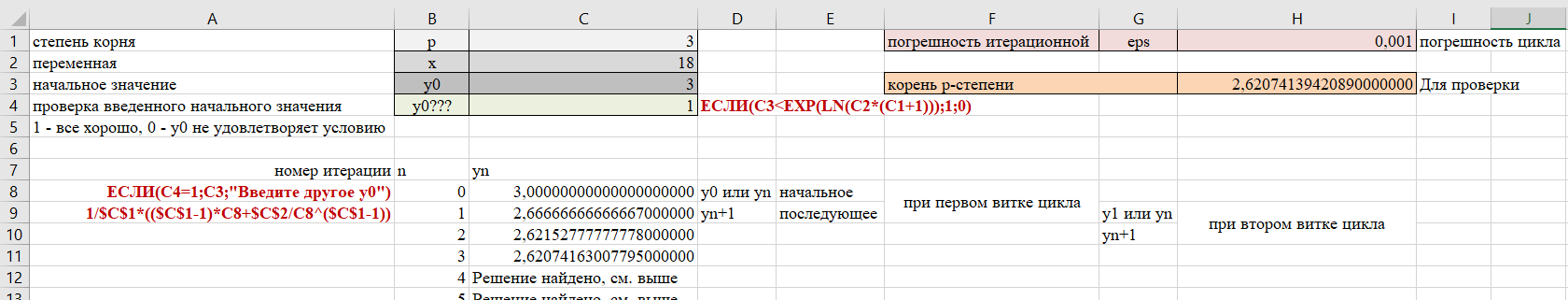


Рисунок 14. Проверка решения задания 9 при  №2

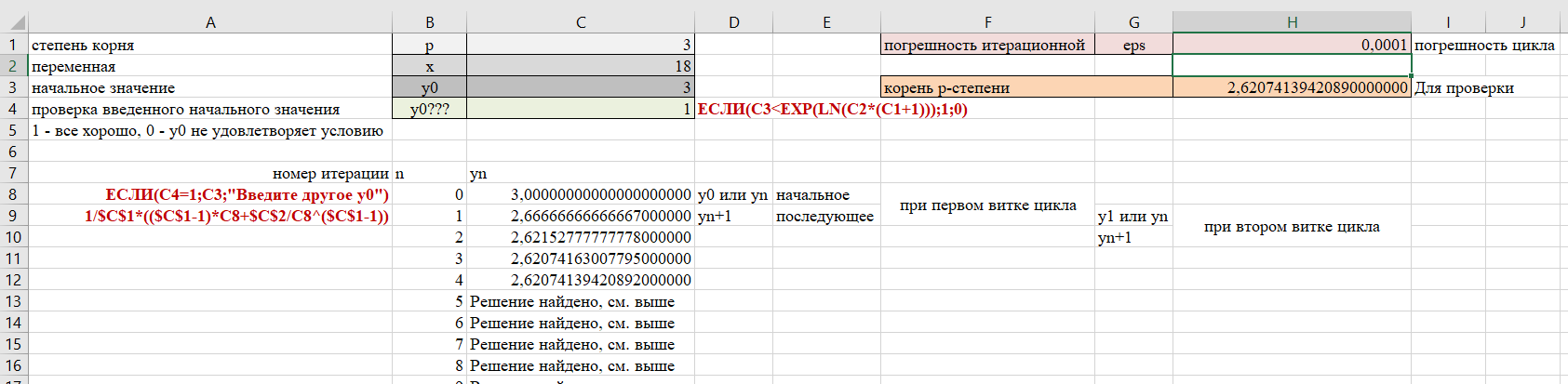


Рисунок 15. Проверка решения задания 9 при  №2

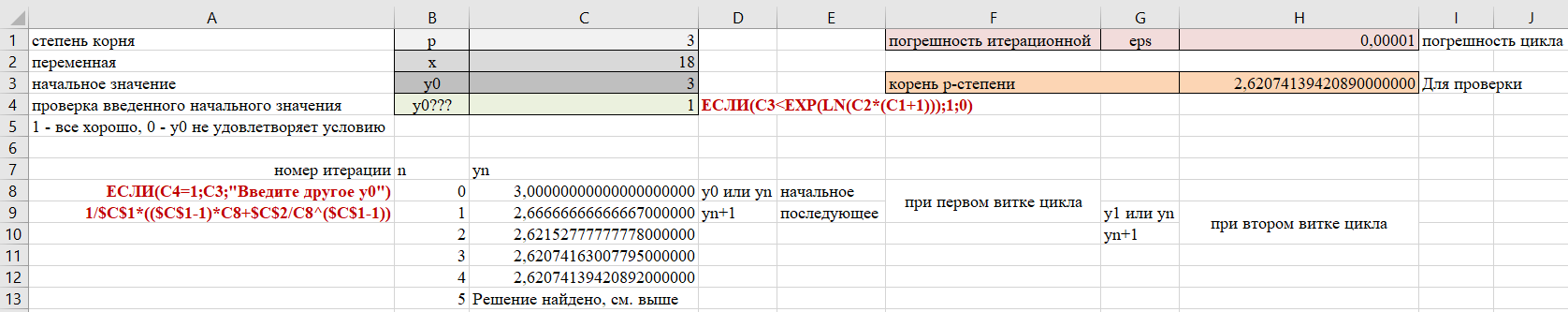


Рисунок 16. Проверка решения задания 9 при  №2

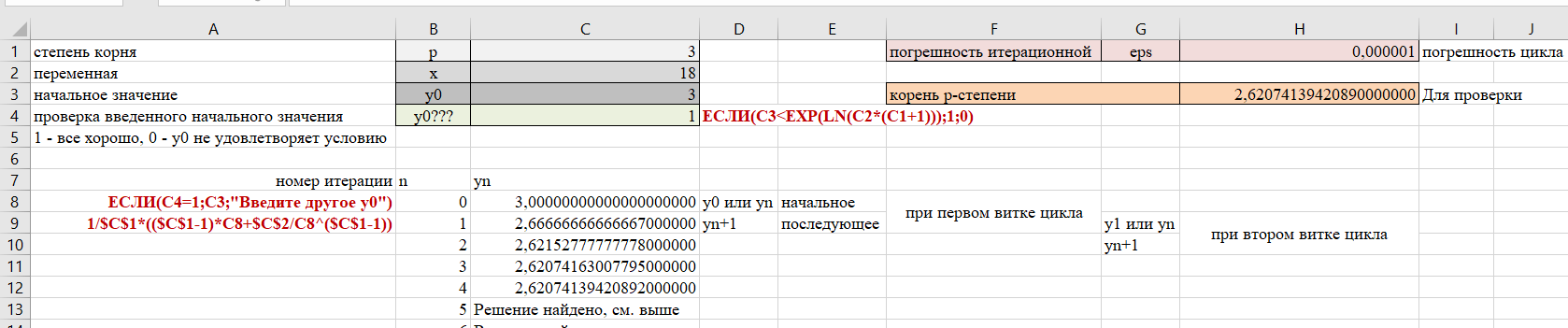


Рисунок 17. Проверка решения задания 9 при  №2

### Проверка №3

На рис.18 представлен вид решения Java

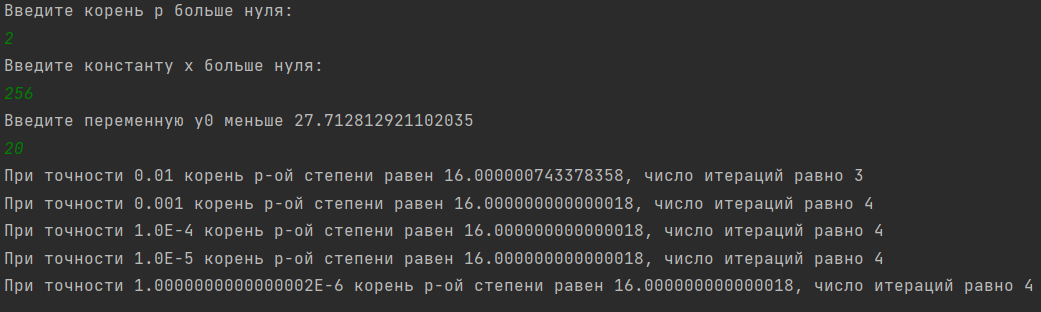


Рисунок 18. Вид решения Java задания 9 №3

На рис. 19-23 представлена проверка задания 9 в MS Excel.

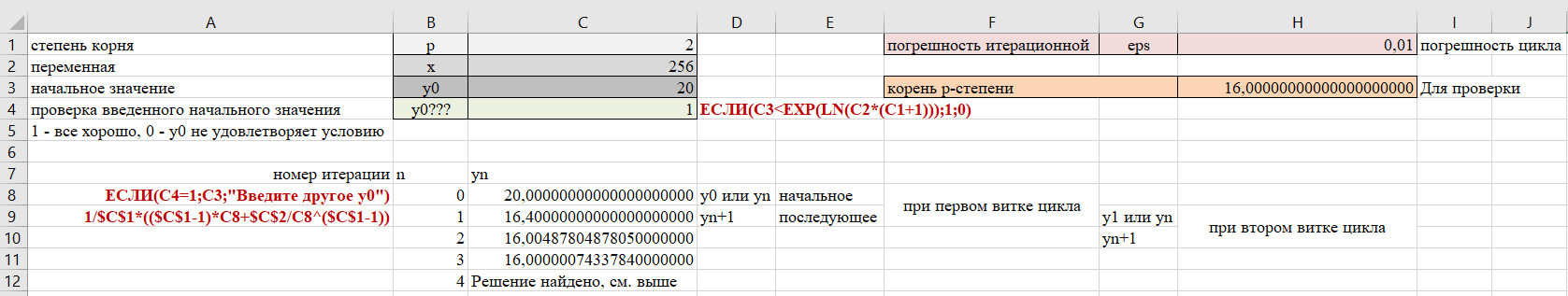


Рисунок 19. Проверка решения задания 9 при  №3

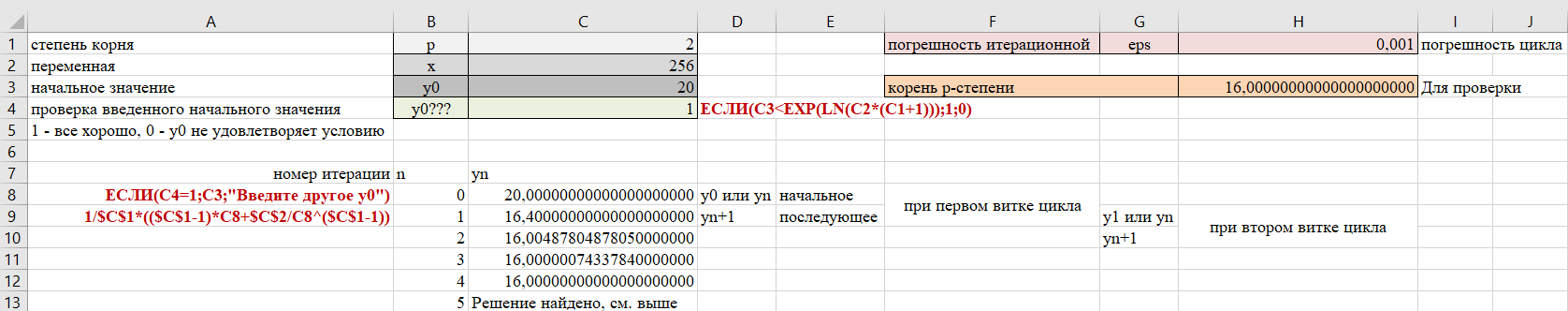


Рисунок 20. Проверка решения задания 9 при  №3

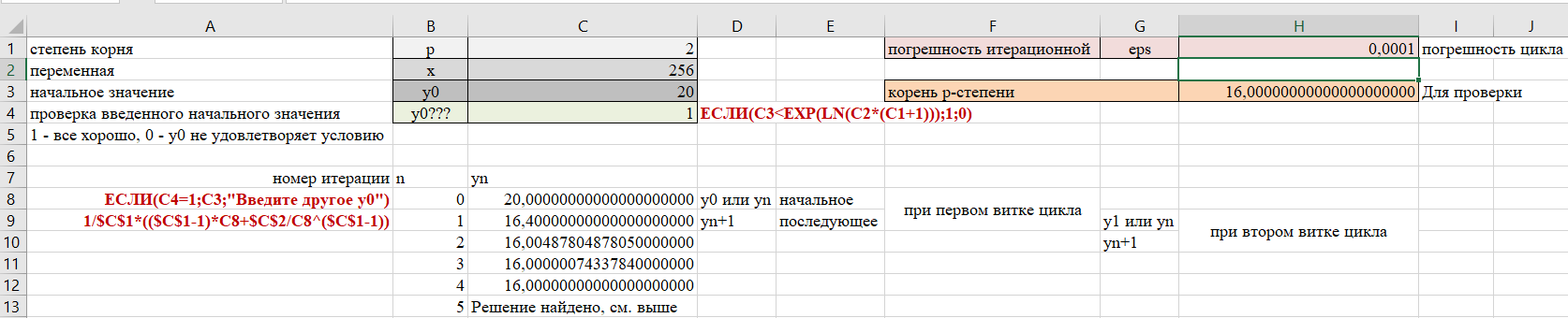


Рисунок 21. Проверка решения задания 9 при  №3

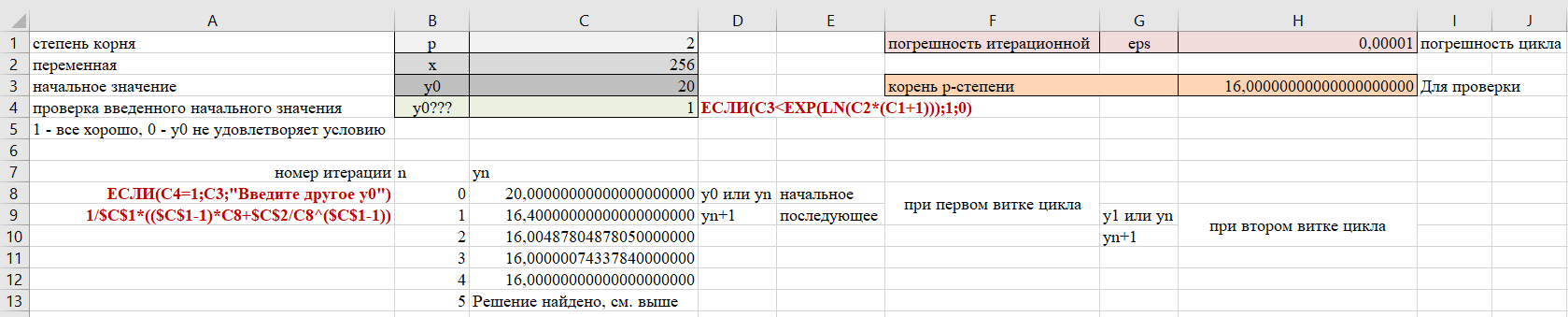


Рисунок 22. Проверка решения задания 9 при  №3

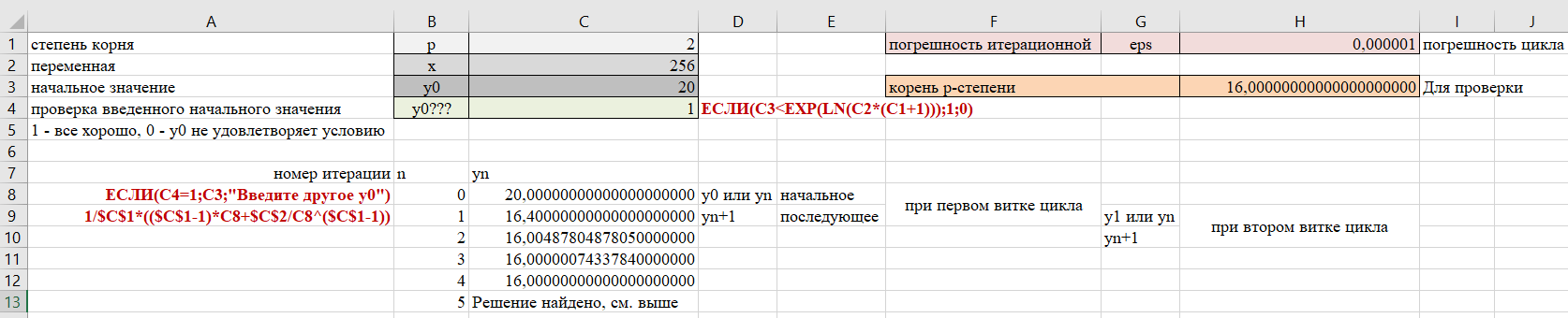


Рисунок 23. Проверка решения задания 9 при  №3

## Блок-схема

-

+

-

Цикл While

-

+

Цикл While

+

-

-

+



Ошибка!

 (\*)



Выход из цикла и вывод всех и количества итераций



, число итераций



, вычисление нового  по (\*)



Ошибка!



Ошибка!

Сравнение решения задачи с помощью трех проверок показала, что решения задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.

# Задание 10

## 10.1. Общая постановка задачи

Разработать алгоритм приближённого вычисления площади криволинейной фигуры, ограниченной осью абсцисс, графиком заданной функции , и вертикальными прямыми, т.е. . Каждый отрезок функции представляется в виде прямоугольника c длиной отрезка (шагом)  (см. рис. ниже). Затем площадь под кривой вычисляется по формуле  – правый прямоугольник, т.к. высота прямоугольника берется как значение функции в крайней правой координате каждого прямоугольника. Вычислить значения площади под кривой при n равном 10, 100, 1000, 10000 в рамках цикла по n. Оценить погрешность решения при разных шагах по сравнению с точным аналитическим решением. Организовать форматированный вывод результатов в виде:

n h S Аналитическое решение Погрешность.

## 10.2. Решение задачи

import static java.lang.System.*out*;  
  
import static java.lang.Math.\*;  
  
public class Task610 {  
 public static void main(String[] args) {  
 *Square*();  
 }  
 public static double Square() {  
 int x = 1;  
 double a = 1;  
 int b = 2;  
 int n = 10;  
 while (n <= 10000) {  
 double h = (b - a) / n;  
 double S = 0;  
 for (int i = 1; i <= n; i++) {  
 S += *Function*(x+i\*h) \* h;  
 }  
 *out*.println("n равно " + n + ", h равно " + h + ", площадь равна " + S +  
 ", результат аналитического решения: " + *Integrate*() +  
 ", погрешность: " + *Error*(S) + " %");  
 n \*= 10;  
 }  
 return 0;  
 }  
 public static double Integrate() {  
 return 8.233732286597172;  
 }  
 public static double Error(double S) {  
 return (*abs*(S - *Integrate*() )/ S \* 100);  
 }  
 public static double Function(double x) {  
 return *cos*(*cos*(*pow*(x, 3) + 5)) + 5 \* x;  
 }  
}

## 10.3. Тестирование с проверкой

Для проверки в MS Excel создана таблица, в которой в таблице A1:D12 в ячейки A2:A12 внесены значения x от 1 до 2 с шагом 0,1, в ячейки B2:B12 записана функция , в ячейки C2:C11 записаны номера итераций i, в ячейки D2:D11 записана площадь S, в D12 сумма площадей. В ячейку B17 записан шаг h, в ячейку A14 записано количество точек на участке графика функции f(x), в ячейке A15 записано количество отрезков, на которые разбивается участок графика функции f(x), в ячейку F19 записано значение площади, полученное аналитическим путем, в ячейку E16 записана формула для вычисления погрешности. В таблицу N1:Q22 аналогично, N2:N22 – x с шагом 0,05, O2:O22 – функция , P2:P21 – номер итерации i, Q2:Q21 – площадь участка S, Q22 – сумма площадей. В ячейку O27 значение шага h, N24 - количество точек на участке графика функции f(x), N25 - количество отрезков, на которые разбивается участок графика функции f(x), Q24 – формула для нахождения погрешности.

E16)=ABS(F19-D12)/F19\*100

Q24)=ABS(F19-Q22)/F19\*100

На рис.24 представлен вид решения в MS Excel.

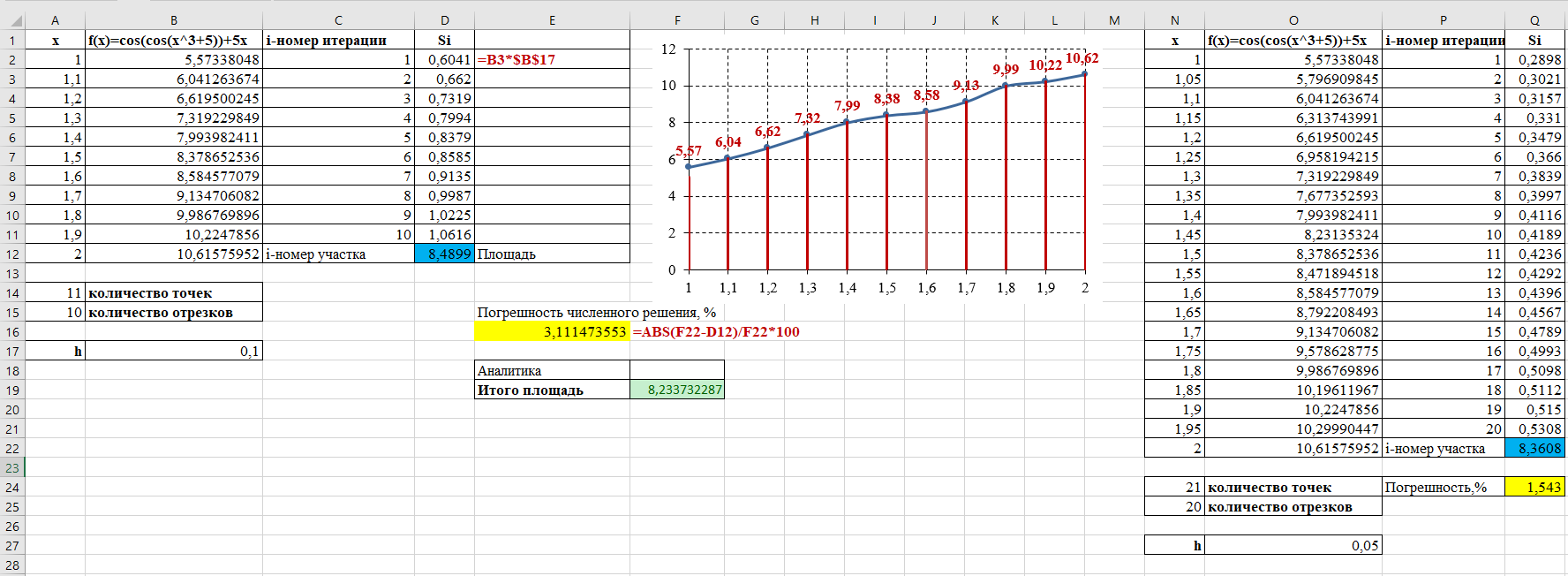


Рисунок 24. Вид решения в MS Excel задания 10

На рис.25 представлен вид решения в Java.

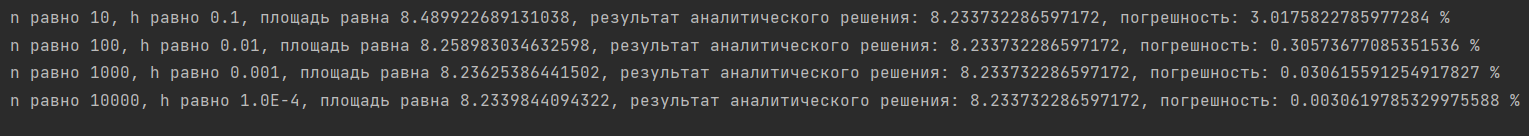


Рисунок 25. Вид решения в Java задания 10



Рисунок 26. График участка функции 

Сравнение решения задачи показало, что решение задачи в Java и MS Excel совпадает. Данный факт подтверждает правильность написанного кода программы. Неопределенностей при решении задачи выявлено не было, возможно данная ситуация связанна с малым количеством проверок данных.